YZTA Bootcamp

Harika, gönderdiğin yapılacaklar listesine (task listesine) göre her bir görevi sana açık ve basit şekilde aşağıda açıklıyorum. Projeniz acil serviste hasta yoğunluğunu azaltmaya yönelik bir yapay zeka destekli sistem. İşte task'ler ve açıklamaları:

**1. Acildeki hasta yoğunluğu azaltmak**

**Ne yapılmalı?**

* Genel hedef bu, tüm sistem bu amacı desteklemeli.
* Bekleme süresini azaltacak yönlendirme, ön değerlendirme (triage) ve otomasyon gibi çözümler üretmelisiniz.
* Yapay zekayla öncelikli hastaların belirlenmesi, yönlendirme önerisi sunulması gibi adımlarla desteklenmeli.

**2. Yeşil - Sarı - Kırmızı alan yönlendirmesi**

**Ne yapılmalı?**

* Hastaların aciliyetine göre sistem tarafından yönlendirilmesini sağlayan bir modül geliştirilmeli.
* Kullanıcının semptomları girildikten sonra yapay zeka aciliyet derecesini (green: hafif, yellow: orta, red: kritik) tahmin eder.
* Bu sınıflandırma için örnek veri seti oluşturabilir veya medikal sınıflandırma kurallarından faydalanabilirsiniz.

**3. Bilgilerini girecek**

**Ne yapılmalı?**

* Kullanıcı adı, soyadı, yaş, cinsiyet, iletişim bilgisi vb. kişisel bilgileri girebileceği bir form ekranı yapılmalı.
* Bu bilgiler veri tabanına kaydedilmeli ve kullanıcıya özel oturum açılmalı (ID ile eşleştirilmeli).

**4. Semptom girecek, öyküsünü girecek kendi dilinden uzunca anlatır**

**Ne yapılmalı?**

* Kullanıcı semptomlarını seçmeli (dropdown/select) ve ardından boş bir kutuya kendince detaylı olarak yazmalı.
* Bu açıklama, doğal dil işleme (NLP) için input olacak.

**5. Argo hasta dilini tıbbi bir dile çevirecek**

**Ne yapılmalı?**

* Kullanıcının yazdığı açıklama, NLP modeline (örneğin GPT veya Türkçe destekli bir LLM) verilip tıbbi terimlere çevrilmeli.
* Örnek: “Kalbim sıkışıyor gibi oluyor” → “Göğüs ağrısı (Angina benzeri semptom)”
* Bunun için basit prompt engineering ya da hazır bir sağlık LLM API'si kullanılabilir.

**6. Raporlayıp doktorun bilgisayarına düşecek**

**Ne yapılmalı?**

* Tıbbi dile çevrilen metin sistem tarafından bir PDF veya ekran bildirimi olarak doktorun paneline yansıtılmalı.
* Web arayüzü varsa doktor giriş ekranı olmalı ve yeni kayıt geldiğinde görebilmeli.

**7. Tüm girilen hastaların semptomları ve sonuçları veri tabanında tutulacak**

**Ne yapılmalı?**

* Giriş yapılan tüm hasta bilgileri, semptomları, AI çıktısı ve yönlendirme önerisi bir veritabanına (PostgreSQL, Firebase vb.) kaydedilmeli.
* Sonuçlarla birlikte doktorun teşhisi de eklenirse modelin eğitimi için çok değerli olur.

**8. Nadir görülen hastalıklar için kayıt tutulacak**

**Ne yapılmalı?**

* Normalden sapmış (örneğin belirli semptom kombinasyonları) veriler özel olarak işaretlenebilir.
* “Nadir durum” olarak bir flag verilebilir ve ayrı bir tabloda kaydedilebilir.
* Bu kayıtlar ileride senaryo üretimi ve eğitim için kullanılacak.

**9. Öğrenci ve hocalar için o nadir semptomlara göre senaryo oluşturulacak**

**Ne yapılmalı?**

* Yukarıdaki nadir vaka kayıtlarından otomatik olarak senaryo çıkaran bir modül yazılmalı.
* Örneğin: “26 yaşında kadın, mide bulantısı, çift görme, baş dönmesi” → eğitim senaryosu haline getirilir.
* Bu senaryo metinleri yapay zeka yardımıyla oluşturulabilir.

**10. Doktor ve öğrencilere bu veriler iletilir**

**Ne yapılmalı?**

* Doktor ve tıp öğrencileri için ayrı bir portal yapılabilir.
* Eğitim sayfası içinde nadir vakalar, senaryolar ve çözüm önerileri yayınlanır.
* Giriş yapan kullanıcı tipine göre filtreleme yapılabilir.

**11. Hasta-doktor kolaylığı ve eğitim artışı**

**Ne yapılmalı?**

* Bu başlık daha çok genel hedef.
* Yukarıdaki tüm sistem parçaları, hem hasta için zaman kazandırır hem de doktor ve öğrenciler için pratik bilgi sağlar.
* Etkiyi göstermek için kullanıcı deneyimi geri bildirimi (feedback formu) alınabilir.

**12. İlaç ve hastalık önerisi**

**Ne yapılmalı?**

* Semptomlara göre olası hastalık tahminleri yapılabilir.
* Bu tahminlere göre genel tedavi yöntemleri veya ilaç türleri önerilebilir.
* Hukuki sınırlamalar nedeniyle doğrudan ilaç ismi vermek yerine “analjezik”, “antibiyotik önerilebilir” gibi genel terimler kullanılmalı.

**Sağlık Alanında AI Proje Görev Listesi - Özet Tablo**

| **Görev Başlığı** | **Ne Yapılmalı?** | **Kullanılabilecek Teknolojiler / Yöntemler** |
| --- | --- | --- |
| **1. Acildeki hasta yoğunluğu azaltmak** | Genel hedef – tüm sistem bu amaç için çalışmalı. | Sistem optimizasyonu, ön değerlendirme algoritması |
| **2. Yeşil/Sarı/Kırmızı alan yönlendirmesi** | Semptomlara göre aciliyet sınıflandırması yapılmalı. | NLP + sınıflandırma modeli (Logistic Regression, Decision Tree), Python, scikit-learn |
| **3. Bilgilerini girecek** | Form ekranı: ad, yaş, cinsiyet vb. bilgileri almalı. | React, HTML/CSS, FastAPI (backend), PostgreSQL (DB) |
| **4. Semptom ve hikaye giriş ekranı** | Kullanıcı semptom seçer ve uzun açıklama yazar. | Form + serbest metin alanı, veri kaydı için backend bağlantısı |
| **5. Argo dili tıbbi dile çeviri** | NLP modeli hasta ifadesini tıbbi dile çevirir. | Türkçe destekli LLM (ör. GPT-4-Turbo), prompt engineering |
| **6. Doktor bilgisayarına rapor** | Tıbbi özet doktor ekranına düşer. | PDF oluşturma (Python'da reportlab), Doktor paneli (React + API) |
| **7. Tüm hasta verilerinin veritabanında tutulması** | Her giriş veritabanında kaydedilir. | PostgreSQL / Firebase, SQLAlchemy |
| **8. Nadir hastalıklar için kayıt** | Sık olmayan semptomlar işaretlenerek kaydedilir. | Nadirlik kontrolü (istatistiksel eşik), ayrı veri tablosu |
| **9. Eğitim senaryosu oluşturma** | Nadir kayıtlar senaryolaştırılır. | Prompt ile otomatik vaka senaryosu üretimi (LLM), markdown veya PDF format |
| **10. Doktor ve öğrenciye iletim** | Portalda senaryolar listelenir. | Kullanıcı giriş sistemli web arayüzü (Next.js / Django) |
| **11. Hasta-doktor kolaylığı ve eğitim artışı** | UX testleri, geri bildirim formları, kullanım kolaylığı hedeflenir. | Google Forms, Session Recording, Hotjar |
| **12. İlaç ve hastalık önerisi** | Semptomlara göre öneri sunulur. | Semptom-hastalık eşleme (knowledge graph veya simple rule-based öneri sistemi) |

**Ek Teknoloji Önerileri (Kılavuza Uyumlu)**

* **LLM/GenAI**: Gemini, GPT-4 Turbo, LLaMA
* **Backend**: FastAPI veya Flask
* **Frontend**: React, Streamlit (hızlı prototipleme için)
* **Database**: PostgreSQL (kararlı), Firebase (hızlı prototip için)
* **Veri görselleştirme**: matplotlib, seaborn, Streamlit components
* **Veri analizi**: pandas, NumPy
* **Hasta verisi standardı (opsiyonel)**: FHIR API